

# Package ‘Phi2rho’

January 20, 2025

**Type** Package

**Title** Owen's T Function and Bivariate Normal Integral

**Version** 1.0.1

**Date** 2023-12-06

**Depends** R (>= 3.5.0), stats

**Imports** Rmpfr

**Description** Computes the Owen's T function or the bivariate normal integral using one of the following: modified Euler's arctangent series, tetrachoric series, or Vasicek's series. For the methods, see Komelj, J. (2023) <[doi:10.4236/ajcm.2023.134026](https://doi.org/10.4236/ajcm.2023.134026)> (or reprint <[arXiv:2312.00011](https://arxiv.org/abs/2312.00011)> with better typography) and Vasicek, O. A. (1998) <[doi:10.21314/JCF.1998.015](https://doi.org/10.21314/JCF.1998.015)>.

**License** GPL-2 | GPL-3

**NeedsCompilation** no

**Author** Janez Komelj [aut, cre]

**Maintainer** Janez Komelj <[jkomelj@siol.net](mailto:jkomelj@siol.net)>

**Repository** CRAN

**Date/Publication** 2023-12-06 21:10:06 UTC

## Contents

Phi2rho-package . . . . .	2
OwenT . . . . .	3
Phi2xy . . . . .	5

<b>Index</b>	<b>10</b>
--------------	-----------

---

Phi2rho-package

*Owen's T Function and Bivariate Normal Integral*

---

## Description

Computes the Owen's T function or the bivariate normal integral.

## Details

The DESCRIPTION file:

Package: Phi2rho

Type: Package

Title: Owen's T Function and Bivariate Normal Integral

Version: 1.0.1

Date: 2023-12-06

Authors@R: person("Janez", "Komelj", role = c("aut", "cre"),

email = "jkomelj@siol.net")

Depends: R (>= 3.5.0), stats

Imports: Rmpfr

Description: Computes the Owen's T function or the bivariate normal integral using one of the following:

modified Euler's arctangent series, tetrachoric series, or Vasicek's series. For the methods,

see Komelj, J. (2023) <doi:10.4236/ajcm.2023.134026>

(or reprint <arXiv:2312.00011> with better typography)

and Vasicek, O. A. (1998) <doi:10.21314/JCF.1998.015>.

License: GPL-2 | GPL-3

## Author(s)

Janez Komelj

Maintainer: Janez Komelj <jkomelj@siol.net>

## References

Komelj, J. (2023): The Bivariate Normal Integral via Owen's T Function as a Modified Euler's Arctangent Series, *American Journal of Computational Mathematics*, **13**, 4, 476–504, doi:10.4236/ajcm.2023.134026 (or reprint <https://arxiv.org/pdf/2312.00011.pdf> with better typography).

Owen, D. B. (1956): Tables for Computing Bivariate Normal Probabilities, *The Annals of Mathematical Statistics*, **27**, 4, 1075–1090, doi:10.1214/aoms/1177728074.

Owen, D. B. (1980): A table of normal integrals, *Communications in Statistics – Simulation and Computation*, **9**, 4, 389–419, doi:10.1080/03610918008812164.

Vasicek, O. A. (1998): A series expansion for the bivariate normal integral, *The Journal of Computational Finance*, **1**, 4, 5–10, doi:10.21314/JCF.1998.015.

**Examples**

```
OwenT(2, 0.5)
OwenT(2, 0.5, fun = "mOwenT") # modified arctangent series (default)
OwenT(2, 0.5, fun = "tOwenT") # tetrachoric series
OwenT(2, 0.5, fun = "vOwenT") # Vasicek's series

rho <- 0.6
a <- rho/sqrt(1 - rho^2)
OwenT(0.3, a)
OwenT(0.3, a, fun = "tOwenT")
OwenT(c(-1, 0.5, 4), a, fun = "vOwenT")

OwenT(2, c(-1, -0.5, 0, 0.5, 1), fun = "vOwenT")

Phi2xy(2, 1.3, 0.5)
Phi2xy(-2, 0.5, -0.3, fun = "tOwenT")
Phi2xy(c(1, 2, -1.5), c(-1, 1, 2.3), 0.5, fun = "vOwenT")
Phi2xy(1, 2, c(-1, -0.5, 0, 0.5, 1))
Phi2xy(c(1, 2), c(-1,3), c(-0.5, 0.8))
```

---

OwenT

*Owen's T Function*


---

**Description**

Computes Owen's T function using the modified Euler's arctangent series, tetrachoric series or Vasicek's series.

**Usage**

```
OwenT(h, a, opt = TRUE, fun = c("mOwenT", "tOwenT", "vOwenT"))
```

**Arguments**

h	Numeric scalar or vector.
a	Numeric scalar or vector.
opt	If TRUE, an optimized calculation is performed.
fun	The name of the internal function being used.

**Details**

If h and a are both vectors, they must be of the same length. If one of h and a is a vector and the other is a scalar, the latter is replicated to the length of the former. The calculation is performed component-wise.

The parameter fun specifies which series is used:

**“mOwenT”**: modified Euler's arctangent series (default).

“**tOwenT**”: tetrachoric series.

“**vOwenT**”: Vasicek’s series.

The `opt` parameter enables checking the results in the submitted article and may be dropped later.

If `fun = "mOwenT"` and `opt = TRUE`, the external arctangent function is used, otherwise all necessary values are calculated on the fly, but usually more iterations are needed.

If `fun = "tOwenT"` or `fun = "vOwenT"`, and `opt = TRUE`, then the parameters transformation is performed when it makes sense, which significantly reduces the number of iterations.

### Value

The value of computed function is returned, scalar or vector. The attribute ‘`nIter`’ of returned value means the number of iterations.

### Note

Function is ready to work with the **Rmpfr** package, which enables using arbitrary precision numbers instead of double precision ones. Assuming **Rmpfr** is loaded, it is sufficient to be called with parameters ‘`h`’ and ‘`a`’, which have class ‘`mpfr`’ and the same precision.

### Author(s)

Janez Komelj

### References

Komelj, J. (2023): The Bivariate Normal Integral via Owen’s T Function as a Modified Euler’s Arc tangent Series, *American Journal of Computational Mathematics*, **13**, 4, 476–504, doi:10.4236/ajcm.2023.134026 (or reprint <https://arxiv.org/pdf/2312.00011.pdf> with better typography).

Owen, D. B. (1956): Tables for Computing Bivariate Normal Probabilities, *The Annals of Mathematical Statistics*, **27**, 4, 1075–1090, doi:10.1214/aoms/1177728074.

Owen, D. B. (1980): A table of normal integrals, *Communications in Statistics – Simulation and Computation*, **9**, 4, 389–419, doi:10.1080/03610918008812164.

### See Also

[Phi2xy](#)

### Examples

```
OwenT(2, 0.5)
OwenT(2, 0.5, fun = "mOwenT") # modified arctangent series (default)
OwenT(2, 0.5, fun = "tOwenT") # tetrachoric series
OwenT(2, 0.5, fun = "vOwenT") # Vasicek's series

rho <- 0.6
a <- rho/sqrt(1 - rho^2)
OwenT(0.3, a)
OwenT(0.3, a, fun = "tOwenT")
```

```

OwenT(c(-1, 0.5, 4), a, fun = "vOwenT")

OwenT(2, c(-1, -0.5, 0, 0.5, 1), fun = "vOwenT")

function (h, a, opt = TRUE, fun = c("mOwenT", "tOwenT", "vOwenT"))
{
  chkArgs1(h = h, a = a, opt = opt)
  if (length(h) < length(a))
    h <- rep(h, length(a))
  if (length(a) < length(h))
    a <- rep(a, length(h))
  z <- h
  n <- rep(0, length(z))
  z[a == 0] <- 0
  z[is.na(a)] <- NA
  i <- a != 0 & !is.na(a)
  if (any(i)) {
    h <- h[i]
    a <- a[i]
    ph <- pnorm(h)
    fun <- match.arg(fun)
    if (fun == "mOwenT")
      j <- abs(a) > 1
    if (fun == "tOwenT")
      j <- opt & abs(a) > 1
    if (fun == "vOwenT")
      j <- opt & abs(a) < 1
    pah <- h
    if (any(j))
      pah[j] <- pnorm(a[j] * h[j])
    pah[is.nan(pah)] <- 0
    w <- eval(call(fun, h, a, ph, pah, opt))
    z[i] <- w
    n[i] <- attr(w, "nIter")
  }
  attr(z, "nIter") <- n
  return(z)
}

```

---

Phi2xy

*Bivariate Normal Integral*


---

### Description

Computes the bivariate normal integral  $\Phi_2(x, y, \rho)$ .

### Usage

```
Phi2xy(x, y, rho, opt = TRUE, fun = c("mOwenT", "tOwenT", "vOwenT"))
```

### Arguments

x	Numeric scalar or vector.
y	Numeric scalar or vector.
rho	Numeric scalar or vector.
opt	If TRUE, an optimized calculation is performed.
fun	The name of the internal function being used.

### Details

The parameter ‘rho’ (or at least one of its components) must be from the interval [-1,1].

Vector parameters must be of the same length, and any scalar parameters are replicated to the same length. The calculation is performed component-wise.

The parameter fun specifies which series is used:

“**mOwenT**”: modified Euler’s arctangent series (default).

“**tOwenT**”: tetrachoric series.

“**vOwenT**”: Vasicek’s series.

The opt parameter enables checking the results in the submitted article and may be dropped later.

If fun = “mOwenT” and opt = TRUE, the external arctangent function is used, otherwise all necessary values are calculated on the fly, but usually more iterations are needed.

If fun = “tOwenT” or fun = “vOwenT”, and opt = TRUE, then the parameters transformation is performed when it makes sense, which significantly reduces the number of iterations.

### Value

The value of computed function is returned, scalar or vector. The attribute ‘nIter’ of returned value means the number of iterations.

### Note

Function is ready to work with the **Rmpfr** package, which enables using arbitrary precision numbers instead of double precision ones. Assuming **Rmpfr** is loaded, it is sufficient to be called with parameters ‘x’, ‘y’ and ‘rho’, which have class ‘mpfr’ and the same precision.

### Author(s)

Janez Komelj

### References

- Komelj, J. (2023): The Bivariate Normal Integral via Owen’s T Function as a Modified Euler’s Arctangent Series, *American Journal of Computational Mathematics*, **13**, 4, 476–504, doi:10.4236/ajcm.2023.134026 (or reprint <https://arxiv.org/pdf/2312.00011.pdf> with better typography).
- Owen, D. B. (1956): Tables for Computing Bivariate Normal Probabilities, *The Annals of Mathematical Statistics*, **27**, 4, 1075–1090, doi:10.1214/aoms/1177728074.
- Owen, D. B. (1980): A table of normal integrals, *Communications in Statistics – Simulation and Computation*, **9**, 4, 389–419, doi:10.1080/03610918008812164.

**See Also**[OwenT](#)**Examples**

```

Phi2xy(2, 1.3, 0.5)
Phi2xy(-2, 0.5, -0.3, fun = "tOwenT")
Phi2xy(c(1, 2, -1.5), c(-1, 1, 2.3), 0.5, fun = "vOwenT")
Phi2xy(1, 2, c(-1, -0.5, 0, 0.5, 1))
Phi2xy(c(1, 2), c(-1,3), c(-0.5, 0.8))

function (x, y, rho, opt = TRUE, fun = c("mOwenT", "tOwenT",
    "vOwenT"))
{
  chkArgs2(x = x, y = y, rho = rho, opt = opt)
  fun <- match.arg(fun)
  sgn <- function(x) {
    y <- sign(x)
    y[y == 0] <- 1
    return(y)
  }
  frx <- function(x, y, rho) {
    rx <- (y - rho * x)/(x * sqrt(1 - rho^2))
    rx[is.nan(rx)] <- 0
    return(-abs(rx) * sgn(y - rho * x) * sgn(x))
  }
  fprx <- function(r, x) {
    u <- r * x
    j <- !is.nan(u)
    if (fun == "mOwenT")
      j <- j & abs(r) > 1
    if (fun == "tOwenT")
      j <- j & opt & abs(r) > 1
    if (fun == "vOwenT")
      j <- j & opt & abs(r) < 1
    u[j] <- pnorm(u[j])
    u[is.nan(u)] <- 0
    return(u)
  }
  fz <- function(x, y, rho, rx, ry, px, py) {
    n <- rep(0, length(x))
    i <- rho != 0 & abs(rho) < 1 & (x != 0 | y != 0)
    z <- x
    if (any(!i)) {
      z[rho == 0] <- px[rho == 0] * py[rho == 0]
      z[rho == +1] <- pmin(px[rho == +1], py[rho == +1])
      z[rho == -1] <- pmax(px[rho == -1] + py[rho == -1] -
        1, 0)
      j <- rho != 0 & abs(rho) < 1
      if (any(j))
        z[j] <- 1/4 + asin(rho[j])/(2 * pi)
    }
  }
}

```

```

    if (any(i)) {
      prx <- fprx(rx[i], x[i])
      pry <- fprx(ry[i], y[i])
      zz <- eval(call(fun, c(x[i], y[i]), c(rx[i], ry[i]),
        c(px[i], py[i]), c(prx, pry), opt, TRUE))
      n[i] <- attr(zz, "nIter")
      z[i] <- (px[i] + py[i])/2 + zz
      j <- i & (x * y < 0 | x * y == 0 & x + y < 0)
      z[j] <- z[j] - 1/2
    }
    attr(z, "nIter") <- n
    return(z)
  }
dim <- max(length(x), length(y), length(rho))
if (isa(x, "mpfr")) {
  pi <- Rmpfr::Const("pi", Rmpfr::getPrec(x))
  z <- mpfrArray(NA, dim = dim, precBits = Rmpfr::getPrec(x))
}
else z <- array(NA, dim = dim)
n <- array(0, dim = dim)
px <- pnorm(x)
py <- pnorm(y)
if (length(x) < dim)
  x <- rep(x, dim)
if (length(y) < dim)
  y <- rep(y, dim)
if (length(rho) < dim)
  rho <- rep(rho, dim)
if (length(px) < dim)
  px <- rep(px, dim)
if (length(py) < dim)
  py <- rep(py, dim)
k <- !is.na(rho) & abs(rho) <= 1
x <- x[k]
y <- y[k]
rho <- rho[k]
px <- px[k]
py <- py[k]
q <- (x^2 - 2 * rho * x * y + y^2)/(2 * (1 - rho^2))
phi <- exp(-q)/(2 * pi * sqrt(1 - rho^2))
phi[is.nan(phi)] <- 0
i <- phi > 1
j <- rho[i] < 0
n1 <- length(rho[i])
n2 <- length(rho) - n1
dim <- 2 * n1 + n2
if (isa(x, "mpfr"))
  xx <- mpfrArray(0, dim = dim, precBits = Rmpfr::getPrec(x))
else xx <- rep(0, dim)
yy <- xx
rr <- xx
pxx <- xx
pyy <- xx

```



```

if (n1 > 0) {
  r <- rho[i]
  u <- x[i]
  v <- y[i] * sgn(r)
  w <- (u - v)/sqrt(2 * (1 - abs(r)))
  r <- -sqrt((1 - abs(r))/2)
  i1 <- 1:(2 * n1)
  xx[i1] <- c(w, -w)
  yy[i1] <- c(v, u)
  rr[i1] <- c(r, r)
  pw <- pnorm(w)
  pv <- (1 - sgn(rho[i]))/2 + sgn(rho[i]) * py[i]
  pxx[i1] <- c(pw, 1 - pw)
  pyy[i1] <- c(pv, px[i])
}
if (n2 > 0) {
  i2 <- (2 * n1 + 1):(2 * n1 + n2)
  xx[i2] <- x[!i]
  yy[i2] <- y[!i]
  rr[i2] <- rho[!i]
  pxx[i2] <- px[!i]
  pyy[i2] <- py[!i]
}
rx <- frx(xx, yy, rr)
ry <- frx(yy, xx, rr)
zz <- fz(xx, yy, rr, rx, ry, pxx, pyy)
nn <- attr(zz, "nIter")
if (n1 > 0) {
  s <- zz[1:n1] + zz[(n1 + 1):(2 * n1)]
  s[j] <- px[i][j] - s[j]
  z[k][i] <- s
  n[k][i] <- nn[1:n1] + nn[(n1 + 1):(2 * n1)]
}
if (n2 > 0) {
  z[k][!i] <- zz[i2]
  n[k][!i] <- nn[i2]
}
attr(z, "nIter") <- n
return(z)
}

```

# Index

- \* **Owen's T function**

  - OwenT, [3](#)

  - Phi2rho-package, [2](#)

- \* **Vasicek's series**

  - Phi2rho-package, [2](#)

- \* **bivariate normal integral**

  - Phi2rho-package, [2](#)

  - Phi2xy, [5](#)

- \* **package**

  - Phi2rho-package, [2](#)

- \* **tetrachoric series**

  - Phi2rho-package, [2](#)

OwenT, [3](#), [7](#)

Phi2rho (Phi2rho-package), [2](#)

Phi2rho-package, [2](#)

Phi2xy, [4](#), [5](#)